END
DATE
FILMED
11-77
DDC

RESOURCE ALLOCATION IN THE PIPELINED SYSTEM

FINAL REPORT

C. V. RAMAMOORTHY

1 JUNE 1973 — 31 DECEMBER 1976

U. S. ARMY RESEARCH OFFICE
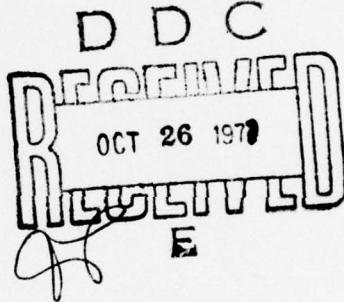
USA-DA-ARO-D-31-124-73-G157

ELECTRONICS RESEARCH LABORATORY
COLLEGE OF ENGINEERING
UNIVERSITY OF CALIFORNIA, BERKELEY 94720

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)	5. TYPE OF REPORT & PERIOD COVERED
Resource allocation in the pipelined system	final 6/1/73 - 12/31/76
	6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s)
C. V. Ramamoorthy	DA-ARO-D-31-124-73-G157

9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Electronics Research Laboratory University of California Berkeley, CA 94720	31 Dec 76

11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE
U. S. Army Research Office Post Office Box 12211 Research Triangle Park, NC 27709	18 p.
	13. NUMBER OF PAGES

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report)
	unclassified
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE NA

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

Final rept. 1 Jun 73 - 31 Dec 76,

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Pipelining, parallel processing, memory organization, modelling, sequencing control, resource allocation, parallel language constructs, parallel execution of sequential programs

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report summarizes the research findings obtained under grant DA-ARO-D-31-123-73-G157. Pipelining, parallel processing and memory organization techniques to improve the capability of computer systems are investigated. In pipelining, results obtained pertain to the modelling, sequencing control, design methods and resource allocation problems of generalized pipeline systems. In parallel processing, language constructs which would constraint the programmer to write parallel programs more amenable to analysis and efficient execution are proposed and a scheme to detect and control the parallel execution of sequentially

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE

(see over)

Abstract (cont.)

organized programs is also developed. Finally, a scheme using intelligent
buffers to improve the performance of interleaved memory is developed and
analyzed.

ACCESSION for

NTIS   White Section
DDC   Buff Section
UNANNOUNCED
JUSTIFICATION

BY
DISTRIBUTION/AVAILABILITY CODES
Dist.   AVAIL

1. Introduction

This report summarizes the findings of our research supported by the U. S. Army Research Office through grant DA-ARO-D-31-124-73-G157. Most of the results reported here are published in the technical papers, Ph.D. dissertations and M.S. theses given in the References. Eleven graduate students have been supported by the grant to pursue advanced degrees at the University of California, Berkeley.

This investigation has been an attempt to establish some foundation for solving design and operational problems related to computer architectures which utilize concurrency in computation to attain more computing power than is allowed by today's electronic technology. The emphasis of our research under the grant has been on pipeline architectures [Ram 77]. But in the course of our research, some new results in parallel processing and memory organization are developed. Parallel processing complement pipelining in increasing the capability of a processing system and the memory bottleneck must be solved if all the advantages of pipelining and parallel processing are to be realized.

We shall briefly discuss our findings in these research areas in three separate sections, one for each topic. A summary section is given at the end.

2. Pipeline Systems

2.1. Overview

Pipelining can be viewed as a mode of exploiting computational parallelism. It can be roughly defined as a technique of imbedding concurrency in execution by constructing a system configuration composed of independent autonomous units each dedicated to perform a specific

-1-

subfunction in an overlapped mode with the others [Ram 74b]. Such an autonomous unit is sometimes called a pipeline segment or facility. A task once initiated flows from segment to segment inside the pipe to be processed. As an illustration, the schematic of a pipelined instruction unit is shown in Fig. 1.

→ | Instruction Fetch 600 nsec | → | Decode 200 nsec | → | Operand Fetch 600 nsec | → | Execution 400 nsec | →
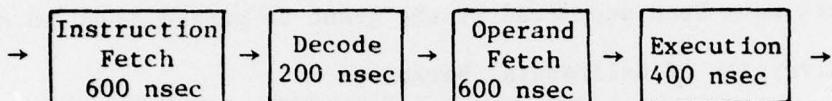
Fig. 1. An example pipelined instruction unit.

It is easy to see that if an uninterrupted stream of instructions is allowed to flow through the pipeline shown in Fig. 1, the throughput rate of the system will be one instruction per 600 nanosecond. The basic objective of pipelining is to maximize throughput through efficient utilization of resources. Compared with parallel processing, pipelining is not geared to shorten response time, but it increases system throughput rate in a more economical way. Furthermore, the effect of overhead on system throughput rate in operating a pipeline system is almost invisible. This is mainly because this overhead can also be overlapped with the other useful operations. Many modern computers (e.g. IBM 360/195, Amdahl 470 and the Cray computers) have extensively utilized pipelining in their processor architecture.

The structure of a pipeline system can be quite complicated. A segment may be used at different times to complete the processing of a task (see Fig. 2). Sometimes there may be several pipelines in the system sharing certain facilities. The TIASC arithmetic unit as shown in Fig. 3 is a notable exampel of such systems. To complicate the issues further, the tasks to be processed may have different processing and resource

-2-

INPUT → S₁ → S₂ → S₃ → S₄

OUTPUT
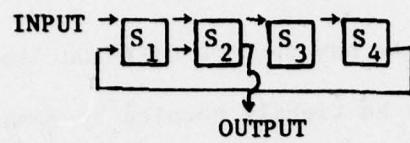
Fig. 2. A single function pipeline with feedback.

FLOATING ADD   FIXED MULT

RECEIVER   REGISTER

EXPONEN SUBTRACT

ALIGN

MULTIPLY

ADD

NORMALIZE

ACCUMULATE

OUTPUT

RESULTS
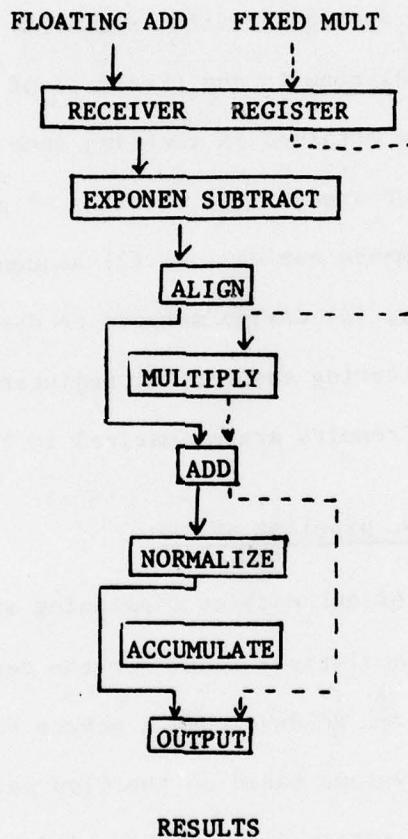
Fig. 3. Pipelined arithmetic unit in TIASC.

An example of multifunction pipeline.

-3-

requirements. The tasks may require different amounts of time to be processed

in a segment; they may be tightly coupled by some precedence relations; and

if not appropriately controlled they may race for the service of shared

segments and resources (e.g. registers). The difficulty of most effectively

designing and controlling the operation of a pipeline system has presented

a very challenging area of fruitful research. We have extensively surveyed

existing theoretical results and practices of pipelining in [Ram 77a].

The results we obtained in tackling some problems in pipelining can

be grouped into four areas: (1) modeling of pipeline systems for effective

control and performance evaluation; (2) sequencing strategies and their

inherent complexity; (3) design methods to optimize cost-effectiveness;

(4) program restructuring and optimal register assignment for pipelined

processors. These results are summarized in the following four sections.

2.2. Modeling of pipeline system

The objective of our earlier pipelining studies was aimed at estab-

lishing an unified analytical model for the design and control of pipeline

systems. In [Red 72], we developed a scheme for categorizing various

types of pipeline systems based on the flow pattern of tasks in the

pipeline, amount of intermediate storage and the nature of the tasks

(e.g. absence or presence of dependence). Subsequently, a study was made

on the various scheduling algorithms for each category [Red 73].

In [Ram 74a], reconfigurable shared resource pipeline (RSRP) systems

were considered in full generality. A useful concept of efficiency

measure which considers cost, speed and space-time span of the segments

of a pipeline system was established so that the effectiveness and

feasibility of a pipeline system can be analyzed. A RSRP system can be

-4-
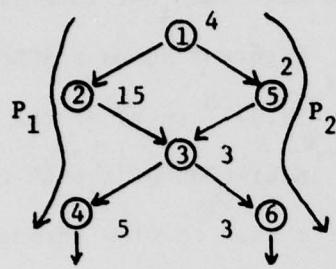
P_1: 1-2-3-4

P_2: 1-5-3-6

(Speed of each facility is as labeled.)

Collision matrix:

$t_{11} = (15, \infty)$

$t_{12} = ((4,10), (16, \infty))$

$t_{21} = (4, \infty)$

$t_{22} = (4, \infty)$

Fig. 4.   Example collision matrix.

modeled by a digraph where a node represents a segment and an arc indicates
the flow of control from one segment to the other. A collision matrix
was introduced in [Ram 74a]. It can be used to avoid collisions inside
a RSRP system. (A collision occurs when more than one task attempts to
use a shared segment at the same time.) Figure 4 shows a RSRP system with
its corresponding collision matrix. The $(i,j)^{th}$ entry of the matrix
represents the time intervals after the initiation of a task to flow
through pipe i so that the excitation of a task to flow through pipe j
will not casue a collision.

Concurrently with the research reported above a scheme called the
dynamic sequencing and segmentation model (DSSM) was developed with the
objective of concealing the run-time overhead required to control the
operation of a general parallel and pipeline system [Ram 74b]. A
simulation study [Ram 73] showed the steady and high performance of the
DSSM and the high feasibility for implementation.

2.3. Sequencing in pipeline systems

In general, the major obstacles to high efficiency in pipeline systems
are caused by: (1) the inherent relationship between the tasks, (2) the
scarcity of some expensive facilities, (3) the unequal and sometimes
unpredictable execution time of the tasks in a given segment; and (4) the
conflicts at some shared resource in the system. These factors indicate
one critical problem to be solved: the optimal sequencing of the tasks to
be admitted into the pipeline system. A sequencing strategy is optimal in
the sense that maximum throughput is achieved while collisions on shared
resources are avoided.

For simple RSRP systems with a bounded queue of ready tasks, an

efficient lookahead scheme to produce locally optimal sequences was developed in [Ram 74a]. The method can be extended to the dynamic case where huge tasks are involved, for example, in a computer network. The inherent complexity of sequencing tasks in a RSRP system was studied in [Ram 75b]. It was found that even under very strong assumptions when we have a static set of ready tasks and the execution time of the pipeline segments are deterministic, the optimal sequencing problem of pipeline system is NP-complete. This implies that its complexity is in the same class as the classical traveling salesman problem. From this result, the semi-exhaustive nature of an optimal strategy for pipeline system sequencing is justified. One implication is that for low level pipeline implementations, faster heuristics are necessary. Some efficient heuristics were proposed in [Li 75] and simulation experiments have demonstrated their effectiveness.

2.5. Design methods

The design problem can be informally stated as the following: Given an application, its throughput, reliability, cost and other requirements, how to most cost-effectively design a pipeline system for the application. In [Li 75], we developed a set of algorithms so that the design problem can be approached in a systematic way rather than using pure instinct, experience and ad hoc solutions. The design of a pipeline system can be approached in the following manner. First a basic skeleton machine and some relevant cost and effect functions can be derived based on the system and application objectives. A set of semi-dynamic programming strategies developed in [Li 75] can then be applied to obtain analytically the most cost-effective pipeline design without exhaustive enumeration. Further by appropriate duplication of some shared resources, a complex RSRP System

-7-

can be partitioned into simplier system. The result of the appropriate
partitioning will lead to reduced control complexity, improved throughput
and reliability. The partitioning algorithms are extended from the
algorithms of Kerninghan-Lin and Ford-Fulkersen. Details of them can
be found in [Li 75]. Another problem of interest is how to introduce
redundancy to a ASRP system for improvement in reliability under cost
constraints. A semi-dynamic programming algorithms for this problem
under weak assumptions was developed also in [Li 75].

It should be mentioned here that these analytical design algorithm
are developed not for the decision making, but as a tool to test decision
and justify prediction or experience. In all cases it is recommended
that both analytical and simulation evaluations be used to search for
a good design.

2.5. Program restructuring and register allocation

The characteristics of a program have significant influence on the
effectiveness and applicability of pipelining. In general, a program
must possess suitable structures as well as abundant parallelism in order
to fully utilize the multi-pipelines available. Restructuring a program
is the process of organizing the original program in such a way that
the final code has "good" characteristics with respect to the system
configuration available. These characteristics would simplify the control
procedure and thus improve the performance of the system. In general,
how to best restructure a program is a difficult problem and it is intimately
related to the pipeline configuration and the sequencing scheme used.
Some efficient program restructuring strategies were developed in [Li 75]
and they are studied, using simulation experiments, in the context of

-8-

some sequencing heuristics on systems whose models were based on some existing machines. (They include the STAR 100 and TIASC pipes).

A closely related problem is how to most effectively assign pipeline resources to tasks. We looked into the special case of assigning registers to instructions to be executed in a pipeline processor. In a pipeline processing system, the operand fetch and preparation phase of one instruction can be overlapped with the actual execution phase of some preceding instructions. While the latter time varies from instruction to instruction, the former is also variable depending on the register assignment. It is crucial for pipeline processors to employ a good register assignment that is not completely insensitive to the processor architecture, otherwise the overlapping power can be severely damaged. This problem is formulated in detail in [Li 77]. In general the problem is also inherently difficult. A non-exhaustive optimal algorithm is found under some strong assumptions. Efficient heuristics with certain performance bounds are proposed for other cases.

3. Parallel Processing

The term parallel processing can be defined as the mode of operation in which different sections of a program are processed by several units of a multi-processing system. The fundamental objective of parallel processing is to execute a program as fast as possible often at all costs which may be incurred. Although parallel processing is quite different from pipelining in the means used to achieve high computing power, these two techniques complement each other in imporving the per-formance of a processing system. Modern computers, almost without exception, utilize both techniques in their architecture. The parallel

-9-

execution unit of the instruction pipeline of the IBM360/91 is a notable example.

Under this grant, we have considered two different approaches to the identification of parallelism in a program. A set of language constructs were developed by which the programmer can explicitly indicate parallelism. On the execution level, a scheme was also developed to detect and control parallelism at run time.

In [Ram. 75a], control parts of parallel programming constructs, one at the machine level and the other at the source level were defined. With respect to these constructs technical foundation is established for detecting useful parallelism hidden in a source parallel program as well as for restructuring a program into the one leading itself to easier analysis and more effective execution. Another objective of these language construct is to impose constraints on the program structure so that the program reliability can be improved. Details of these constructs can be found in [Kim 74].

More recently, we developed a scheme to detect and control the execution of parallel tasks in run time [Ram. 76]. Unlike the traditional lookahead approach, parallelism detection is done by controls local to the processors in the new scheme. With information about how variables are used in the program and the status of tasks being executed, these local controls cooredinate with each other and synchronize the action of the processors so that precedence among tasks are preserved. The scheme minimized the overhead time before parallel execution and thus in effect removed the parallelism detection procedure as a bottleneck of the parallel processing system.

Associative search methods are also studied. An algorithm for ordered retrieval was developed in [Ram. 77b]. It is believed that the algorithm is the best one ever presented.

4. Memory Organization

The problem of memory contention has a significant effect on the efficiency of any pipeline and/or parallel system. This problem occurs when more than one task being executed concurrently needs to access the same memory module.

A scheme using intelligent buffers was developed for an interleaved memory in [Wah 76]. The goal there is to improve the performance of the memory by the addition of a small number of buffers. An analytical model based on discrete Markov chains has been developed to evaluate the scheme. The results show that significant improvement can be achieved with a small number of buffers. The analytical result is verified by a trace driven simulation.

5. Summary

This report summarizes the research findings obtained under grant DA-ARO-D-31-123-73-G157. Three related areas of advanced computer architecture are investigated. In pipelining, results obtained pertain to the modelling, sequencing control, design methods and tasks resource allocation problems of generalized pipeline systems. In parallel processing, language constructs which can effectively identify parallel tasks and a new scheme to detect parallelism in run time are developed. An efficient associative search algorithm for ordered retrieval is also proposed. Finally, a scheme using intelligent buffers to improve the performance of interleaved memory is developed and analyzed.

REFERENCES

[Kim 74]   Kim, K.H., "Optimizing Architecture in Parallel Processing,"
            Ph.D. Thesis, Mem. No. ERL-M482, U.C. Berkeley, Nov. 1974.

[Leu 75]   Leung, W.H., "A New Approach to the Parallel Processing of
            Sequential Programs," M.S. Thesis, U.C. Berkeley, 1975.

[Li 75]   Li, H.F., "A Structured Study of Parallel Pipelined Systems,"
            Ph.D. Thesis, Mem. No. ERL-M530, U.C. Berkeley, Aug. 1975.

[Li 77]   Li, H.F., "Register Assignment in Overlapped Processing
            Systems," To appear.

[Ram 73]   Ramamoorthy, C.V. and Kim, K.H., "Dynamic Sequencing and
            Segmentation in the Generalized Pipelining System," Proc. 2nd
            Texas Conference on Computing Systems, Austin, Texas, Nov. 1973.

[Ram 74a]   Ramamoorthy, C.V. and Li, H.F., "Efficiency in Generalized
            Pipeline Networks," AFIPS, Vol. 43, 1974.

[Ram 74b]   Ramamoorthy, C.V. and Kim, K.H., "Pipelining — the Generalized
            Concept and Sequencing Strategies," AFIPS, 1974.

[Ram 75a]   Ramamoorthy, C.V. and Kim, K.H., "A Method of Structuring
            and Validating Parallel Programs," Compcon Spring, 1975.

[Ram 75b]   Ramamoorthy, C.V. and Li, H.F., "Sequencing Control in
            Multifunctional Pipeline Systems," Sagamore Computer
            Conference, 1975.

[Ram 75c]   Ramamoorthy, C.V. and Li, H.F., "Pipeline Processors — A
            Survey," Sagamore Computer Conference, 1975.

[Ram 75d]   Ramamoorthy, C.V. and Kim, K.H., "A Method of Structuring
             and Validating Parallel Programs," Proc. Compcon 1975.

[Ram 75e]   Ramamoorthy, C.V and Li, H.F., "The Design Operations
             and Performance of a Multiprocessor System," INFOTECH
             Series on Multiprocessors, July, 1975.

[Ram 76a]   Ramamoorthy, C.V. and Leung, W.H., "A Scheme for the
             Parallel Execution of Sequential Programs," Proc. International
             Conference on Parallel Processing, 1976.

[Ram 76b]   Ramamoorthy, C.V. and Krishnarao, T., "Software-Hardware
             Support for the Application of Microprocessors," 7th
             International Congress on Microelectronics, 1976.

[Ram 77a]   Ramamoorthy, C.V. and Li, H.F., "Pipeline Architecture,"
             ACM Computer Surveys, March 1977.

[Ram 77b]   Ramamoorthy, C.V., Turner, J.L. and Wah, B.W., "A Design
             of a Fast Sorting Associative Memory," to appear in IEEETC.

[Red 72]   Reddi, S. and Ramamoorthy, C.V., "Sequencing Strategies in
             Pipeline Computer Systems," Technical Report No. 134,
             Information Systems Research Labotatory, U.T. at Austin,
             Aug. 1972.

[Red 73]   Reddi, S.S. and Ramamoorthy, C.V., "A Scheduling Problem,"
             Operation Research Quarterly, Vol. 24, No. 3, Sept. 1973.

[Wah 76]   Wah, B.W., "The Analysis of Buffering in an Interleaved
             Memory System," M.S. Report, U.C. Berkeley, 1976.

-13-

Personnels Supported   (all grad students in Berkeley)

I.   Ph.D.

(1) O. Alton   Dec. 1974

(2) H.F. Li   Aug. 1975

(3) K.H. Kim   Nov. 1974

II. M.S.

(1) I. Wendel   Dec. 1974

(2) A.C. Yao   Nov. 1976

(3) F.S. B. Ho   June 1976

(4) T. Krishnarao   Aug. 1975

(5) W.H. Leung   Dec. 1975

(6) B.W. Wah   Dec. 1976

III. Others

(1) G.S. Ho   candidate for M.S.